2.Create representation of document by calculating Term Frequency and Inverse Document Frequency.

term frequency(TF):

In [1]:
```
 #import the necessary libraries
import pandas as pd
import sklearn as sk
import math
```

In [2]:
```
first_sentence = "Data Science is the sexiest job of the 21st century"
second_sentence = "machine learning is the key for data science"
third_sentence = "machine learning is part of artificial intelligence"
fourth_sentence= "A data scientist has to think more than code "
fifth_sentence = "artificial intelligance is an emerging and promising technology
#split so each word have their own string
first_sentence = first_sentence.split(" ")
second_sentence = second_sentence.split(" ")
third_sentence = third_sentence.split(" ")
fourth_sentence = fourth_sentence.split(" ")
fifth_sentence = fifth_sentence.split(" ")
#join them to remove common duplicate words
total= set(first_sentence).union(set(second_sentence).union(third_sentence).unior
print(total)
```

```
{'', 'key', 'century', 'learning', 'A', 'of', 'for', 'an', 'Data', 'job', 'dat
a', 'to', 'is', 'and', 'than', 'machine', 'has', 'sexiest', '21st', 'the', 'Sci
ence', 'science', 'think', 'part', 'code', 'promising', 'intelligance', 'more',
'emerging', 'scientist', 'intelligence', 'technology', 'artificial'}
```

In [3]:
```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
wordDictC = dict.fromkeys(total, 0)
wordDictD = dict.fromkeys(total, 0)
wordDictE = dict.fromkeys(total, 0)
for word in first_sentence:
    wordDictA[word]+=1

for word in second_sentence:
    wordDictB[word]+=1

for word in third_sentence:
    wordDictC[word]+=1

for word in fourth_sentence:
    wordDictD[word]+=1

for word in fifth_sentence:
    wordDictE[word]+=1
```

In [4]:
```
pd.DataFrame([wordDictA, wordDictB, wordDictC, wordDictD, wordDictE])
```

Out[4]:

	key	century	learning	A	of	for	an	Data	job	...	part	code	promising	intelligance	m
0	0	0	1	0	0	1	0	0	1	1	...	0	0	0	0	
1	0	1	0	1	0	0	1	0	0	0	...	0	0	0	0	
2	0	0	0	1	0	1	0	0	0	0	...	1	0	0	0	
3	1	0	0	0	1	0	0	0	0	0	...	0	1	0	0	
4	0	0	0	0	0	0	0	1	0	0	...	0	0	1	1	

5 rows × 33 columns

In [5]:
```
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)
    for word, count in wordDict.items():
        tfDict[word] = count/float(corpusCount)
    return(tfDict)
#running our sentences through the tf function:
tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)
tfThird = computeTF(wordDictC, third_sentence)
tfFourth = computeTF(wordDictD, fourth_sentence)
tfFifth = computeTF(wordDictE, fifth_sentence)
```

In [6]:
```
#Converting to dataframe for visualization
tf =pd.DataFrame([tfFirst, tfSecond, tfThird,tfFourth, tfFifth])
print(tf)
```

```
          key  century  learning    A        of    for     an  Data  job  ...
\
0  0.0  0.000      0.1  0.000000  0.0  0.100000  0.000  0.000   0.1  0.1  ...
1  0.0  0.125      0.0  0.125000  0.0  0.000000  0.125  0.000   0.0  0.0  ...
2  0.0  0.000      0.0  0.142857  0.0  0.142857  0.000  0.000   0.0  0.0  ...
3  0.1  0.000      0.0  0.000000  0.1  0.000000  0.000  0.000   0.0  0.0  ...
4  0.0  0.000      0.0  0.000000  0.0  0.000000  0.000  0.125   0.0  0.0  ...

       part  code  promising  intelligance  more  emerging  scientist  \
0  0.000000   0.0      0.000         0.000   0.0     0.000        0.0
1  0.000000   0.0      0.000         0.000   0.0     0.000        0.0
2  0.142857   0.0      0.000         0.000   0.0     0.000        0.0
3  0.000000   0.1      0.000         0.000   0.1     0.000        0.1
4  0.000000   0.0      0.125         0.125   0.0     0.125        0.0

   intelligence  technology  artificial
0      0.000000       0.000    0.000000
1      0.000000       0.000    0.000000
2      0.142857       0.000    0.142857
3      0.000000       0.000    0.000000
4      0.000000       0.125    0.125000

[5 rows x 33 columns]
```

In [7]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
filtered_sentence = [w for w in wordDictA if not w in stop_words]
print(filtered_sentence)
```

```
['', 'key', 'century', 'learning', 'A', 'Data', 'job', 'data', 'machine', 'sexi
est', '21st', 'Science', 'science', 'think', 'part', 'code', 'promising', 'inte
lligance', 'emerging', 'scientist', 'intelligence', 'technology', 'artificial']

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\RAKSHANDA\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [8]:
```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)

    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))

    return(idfDict)
#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB, wordDictC, wordDictD, wordDictE])
```

In [9]:
```
def computeTFIDF(tfBow, idfs):
    tfidf = {}
    for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
    return(tfidf)
#running our two sentences through the IDF:
idfFirst = computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)
idfThird = computeTFIDF(tfThird, idfs)
idfFourth = computeTFIDF(tfFourth, idfs)
idfFifth = computeTFIDF(tfFifth, idfs)
#putting it in a dataframe
idf= pd.DataFrame([idfFirst, idfSecond, idfThird, idfFourth, idfFifth])
print(idf)
```

```
                 key   century  learning         A        of       for  \
0  0.000000  0.000000  0.069897  0.000000  0.000000  0.069897  0.000000
1  0.000000  0.087371  0.000000  0.087371  0.000000  0.000000  0.087371
2  0.000000  0.000000  0.000000  0.099853  0.000000  0.099853  0.000000
3  0.069897  0.000000  0.000000  0.000000  0.069897  0.000000  0.000000
4  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000

         an      Data       job  ...      part      code  promising  \
0  0.000000  0.069897  0.069897  ...  0.000000  0.000000   0.000000
1  0.000000  0.000000  0.000000  ...  0.000000  0.000000   0.000000
2  0.000000  0.000000  0.000000  ...  0.099853  0.000000   0.000000
3  0.000000  0.000000  0.000000  ...  0.000000  0.069897   0.000000
4  0.087371  0.000000  0.000000  ...  0.000000  0.000000   0.087371

   intelligance      more  emerging  scientist  intelligence  technology  \
0      0.000000  0.000000  0.000000   0.000000      0.000000    0.000000
1      0.000000  0.000000  0.000000   0.000000      0.000000    0.000000
2      0.000000  0.000000  0.000000   0.000000      0.099853    0.000000
3      0.000000  0.069897  0.000000   0.069897      0.000000    0.000000
4      0.087371  0.000000  0.087371   0.000000      0.000000    0.087371

   artificial
0    0.000000
1    0.000000
2    0.099853
3    0.000000
4    0.087371

[5 rows x 33 columns]
```

In [ ]:
```
```